A Simple Semi-supervised Algorithm For Named Entity Recognition

Wenhui Liao and Sriharsha Veeramachaneni
Research and Develpment,Thomson Reuters
610 Opperman Drive, Eagan MN 55123
{wenhui.liao, harsha.veeramachaneni}@thomsonreuters.com

Abstract

We present a simple semi-supervised learning algorithm for named entity recognition (NER) using conditional random fields (CRFs). The algorithm is based on exploiting evidence that is independent from the features used for a classifier, which provides high-precision labels to unlabeled data. Such independent evidence is used to automatically extract high-accuracy and non-redundant data, leading to a much improved classifier at the next iteration. We show that our algorithm achieves an average improvement of 12 in recall and 4 in precision compared to the supervised algorithm. We also show that our algorithm achieves high accuracy when the training and test sets are from different domains.

1 Introduction

Named entity recognition (NER) or tagging is the task of finding names such as organizations, persons, locations, etc. in text. Since whether or not a word is a name and the entity type of a name are determined mostly by the context of the word as well as by the entity type of its neighbors, NER is often posed as a sequence classification problem and solved by methods such as hidden Markov models (HMM) and conditional random fields (CRF).

Automatically tagging named entities (NE) with high precision and recall requires a large amount of hand-annotated data, which is expensive to obtain. This problem presents itself time and again because tagging the same NEs in different domains usually requires different labeled data. However, in most domains one often has access to large amounts of unlabeled text. This fact motivates semi-supervised approaches for NER.

Semi-supervised learning involves the utilization of unlabeled data to mitigate the effect of insufficient labeled data on classifier accuracy. One variety of semi-supervised learning essentially attempts to automatically generate high-quality training data from an unlabeled corpus. Algorithms such as co-training (Blum and Mitchell, 1998)(Collins and Singer, 1999)(Pierce and Cardie, 2001) and the Yarowsky algorithm (Yarowsky, 1995) make assumptions about the data that permit such an approach.

The main requirement for the automatically generated training data in addition to high accuracy, is that it covers regions in the feature space with low probability density. Furthermore, it is necessary that all the classes are represented according to their prior probabilities in every region in the feature space. One approach to achieve these goals is to select unlabeled data that has been classified with low confidence by the classifier trained on the original training data, but whose labels are known with high precision from *independent* evidence. Here independence means that the high-precision *decision rule* that classifies these low confidence instances uses information that is independent of the features used by the classifier.

We propose two ways of obtaining such independent evidence for NER. The first is based on the fact that multiple mentions of capitalized tokens are likely to have the same label and occur in independently chosen contexts. We call this the

Proceedings of the NAACL HLT Workshop on Semi-supervised Learning for Natural Language Processing, pages 58–65,
Boulder, Colorado, June 2009.

multi-mention property. The second is based on the fact that entities such as organizations, persons, etc., have context that is highly indicative of the class, yet is independent of the other context (e.g. company suffixes like Inc., Co., etc., person titles like Mr., CEO, etc.). We call such context *high precision independent context*.

Let us first look at two examples.

Example 1:

1) said Harry You, CEO of HearingPoint ....

2) For this year's second quarter, You said the company's ...

The classifier tags "Harry You" as person (*PER*) correctly since its context (said, CEO) makes it an obvious name. However, in the second sentence, the classifier fails to tag "You" as a person since "You" is usually a stopword. The second sentence is exactly the type of data needed in the training set.

Example 2:

(1) Medtronic Inc 4Q profits rise 10 percent...

(2) Medtronic 4Q profits rise 10 percent...

The classifier tags "Medtronic" correctly in the first sentence because of the company suffix "Inc" while it fails to tag "Medtronic" in the second sentence since "4Q profits" is a new pattern and "Medtronic" is unseen in the training data. Thus the second sentence is what we need in the training set.

The two examples have one thing in common. In both cases, the second sentence has a new pattern and incorrect labels, which can be fixed by using either multi-mention or high-precision context from the first sentence. We actually *artificially* construct the second sentence to be added to the training set in Example 2 although only the first sentence exists in the unlabeled corpus.

By leveraging such independent evidence, our algorithm can automatically extract high-accuracy and non-redundant data for training, and thus obtain an improved model for NER. Specifically, our algorithm starts with a model trained with a small amount of gold data (manually tagged data). This model is then used to extract high-confidence data, which is then used to discover low-confidence data by using other independent features. These low-confidence data are then added to the training data to retrain the model. The whole process repeats until no significant improvement can be achieved. Our experiments show that the algorithm is not only much better than the initial model, but also better than the supervised learning when a large amount of gold data are available. Especially, even when the domain from which the original training data is sampled is different from the domain of the testing data, our algorithm still provides significant gains in classification accuracy.

2 Related Work

The Yarowsky algorithm (Yarowsky, 1995), originally proposed for word sense disambiguation, makes the assumption that it is very unlikely for two occurrences of a word in the same discourse to have different senses. This assumption is exploited by selecting words classified with high confidence according to sense and adding other contexts of the same words in the same discourse to the training data, even if they have low confidence. This allows the algorithm to learn new contexts for the senses leading to higher accuracy. Our algorithm also uses multi-mention features. However, the application of the Yarowsky algorithm to NER involves several domain-specific choices as will become evident below.

Wong and Ng (Wong and Ng, 2007) use the same idea of multiple mentions of a token sequence being to the same named entity for feature engineering. They use a named entity recognition model based on the maximum entropy framework to tag a large unlabeled corpus. Then the majority tags of the named entities are collected in lists. The model is then retrained by using these lists as extra features. This method requires a sufficient amount of manually tagged data initially to work. Their paper shows that, if the initial model has a low F-score, the model with the new features leads to low F-score too. Our method works with a small amount of gold data because, instead of constructing new features, we use independent evidence to enrich the training data with high-accuracy and non-redundant data.

The co-training algorithm proposed by Blum and Mitchell (Blum and Mitchell, 1998) assumes that the features can be split into two class-conditionally independent sets or "views" and that each view is sufficient for accurate classification. The classifier built on *one* of the views is used to classify a large unlabeled corpus and the data classified with high-

confidence are added to the training set on which the classifier on the *other* view is trained. This process is iterated by interchanging the views. The main reason that co-training works is that, because of the class-conditional independence assumptions, the high-confidence data from one view, in addition to being highly precise, is unbiased when added to the training set for the other view. We could not apply co-training for semi-supervised named entity recognition because of the difficulty of finding informative yet class-conditionally independent feature sets.

Collins et al.(Collins and Singer, 1999) proposed two algorithms for NER by modifying Yarowsky's method (Yarowsky, 1995) and the framework suggested by (Blum and Mitchell, 1998). However, all their features are at the word sequence level, instead of at the token level. At the token level, the seed rules they proposed do not necessarily work. In addition, parsing sentences into word sequences is not a trivial task, and also not necessary for NER, in our opinion.

Jiao et al. propose semi-supervised conditional random fields (Jiao et al., 2006) that try to maximize the conditional log-likelihood on the training data and simultaneously minimize the conditional entropy of the class labels on the unlabeled data. This approach is reminiscent of the semi-supervised learning algorithms that try to discourage the boundary from being in regions with high density of unlabeled data. The resulting objective function is no longer convex and may result in local optima. Our approach in contrast avoids changing the CRF training procedure, which guarantees global maximum.

3 Named Entity Recognition

As long as independent evidence exists for one type of NE, our method can be directly applied to classify such NE. As an example, we demonstrate how to apply our method to classify three types of NEs: organization (ORG), person (PER), and location (LOC) since they are the most common ones. A non-NE is annotated as O.

3.1 Conditional Random Fields for NER

We use CRF to perform classification in our framework. CRFs are undirected graphical models trained to maximize the conditional probability of a sequence of labels given the corresponding input sequence. Let X, $X = x_1...x_N$, be an input sequence, and Y, $Y = y_1....y_N$, be the label sequence for the input sequence. The conditional probability of Y given X is:

$$P(Y|X) = \frac{1}{Z(X)} \exp(\sum_{n=1}^{N} \sum_{k} \lambda_k f_k(y_{n-1}, y_n, X, n)) \quad (1)$$

where $Z(X)$ is a normalization term, f_k is a feature function, which often takes a binary value, and λ_k is a learned weight associated with the feature f_k. The parameters can be learned by maximizing log-likelihood ℓ which is given by

$$\ell = \sum_{i} \log P(Y_i|X_i) - \sum_{k} \frac{\lambda_k^2}{2\sigma_k^2} \quad (2)$$

where σ_k^2 is the smoothing (or regularization) parameter for feature f_k. The penalty term, used for regularization, basically imposes a prior distribution on the parameters.

It has been shown that ℓ is convex and thus a global optimum is guaranteed (McCallum, 2003). Inferring label sequence for an input sequence X involves finding the most probable label sequence, $Y^* = \arg\max_{Y} P(Y|X)$, which is done by the Viterbi algorithm (Forney, 1973).

3.2 Features for NER

One big advantage of CRF is that it can naturally represent rich domain knowledge with features.

3.2.1 Standard Features

Part of the features we used for our CRF classifier are common features that are widely used in NER (McCallum and Li, 2003), as shown below.

1) *Lexicon*. Each token is itself a feature.

2) *Orthography*. Orthographic information is used to identify whether a token is capitalized, or an acronym, or a pure number, or a punctuation, or has mixed letters and digits, etc.

3) *Single/multiple-token list*. Each list is a collection of words that have a common sematic meaning, such as last name, first name, organization, company suffix, city, university, etc.

4) *Joint features*. Joint features are the conjunctions of individual features. For example, if a token is in a last name list and its previous token is in a title list, the token will have a joint feature called as Title+Name.

5) *Features of neighbors*. After extracting the above features for each token, its features are then copied to its neighbors (The neighbors of a token include the previous two and next two tokens) with a position id. For example, if the previous token of a token has a feature "Cap@0", this token will have a feature "Cap@-1".

3.2.2 Label Features

One unique and important feature used in our algorithm is called *Label Features*. A label feature is the output label of a token itself if it is known. We designed some simple high-precision rules to classify each token, which take precedence over the CRF. Specifically, if a token does not include any uppercase letter, is not a number, and it is not in the *nocap* list (which includes the tokens that are not capitalized but still could be part of an NE, such as al, at, in, -, etc), the label of this token is "O".

Table 1: An example of extracted features

Tokens	**Feature**
Monday	W=Monday@0 O@0
vice	W=vice@0 O@0
chairman	W=chairman@0 title@0 O@0
Goff	W=Goff@0 CAP@0 Lastname@0 W=chairman@-1 title@-1 O@-1 W=vice@-2 O@-2 W=said@1 O@1 W=it@2 O@2
said	W=said@0 O@0
the	W=it@0 O@0
company	W=company@0 O@0

In addition, if a token is surrounded by "O" tokens and is in a *Stopword* list, or in a *Time* list (a collection of date, time related tokens), or in a *nocap* list, or a *nonNE* list (a collection of tokens that are unlikely to be an NE), or a pure number, its label is "O" as well. For example, in the sentence "Ford has said there is no plan to lay off workers", all the tokens except "Ford" have "O" labels. More rules can be designed to classify NE labels. For example, if a token is in an unambiguousORG list, it has a label "ORG".

For any token with a known label, unless it is a neighbor of a token with its label unknown (i.e., not pretagged with high precision), its features include only its lexicon and its label itself. No features will be copied from its neighbors either. Table 1 gives an example to demonstrate the features used in our algorithm. For the sentence "Monday vice chairman Goff said the company ...", only "Goff" includes its own features and features copied from its neighbors, while most of the other tokens have only two features since they are "O" tokens based on the high-precision rules.

Usually, more than half the tokens will be classified as "O". This strategy greatly saves feature extraction time, training time, and inference time, as well as improving the accuracy of the model. Most importantly, this strategy is necessary in the semi-supervised learning, which will be explained in the next section.

4 Semi-supervised Learning Algorithm

Our semi-supervised algorithm is outlined in Table 2. We assume that we have a small amount of labeled data L and a classifier C_k that is trained on L. We exploit a large unlabeled corpus U from the test domain from which we automatically and gradually add new training data D to L, such that L has two properties: 1) *accurately labeled*, meaning that the labels assigned by automatic annotation of the selected unlabeled data are correct, and 2) *non-redundant*, which means that the new data is from regions in the feature space that the original training set does not adequately cover. Thus the classifier C_k is expected to get better monotonically as the training data gets updated.

Table 2: The semi-supervised NER algorithm

Given:
- L - a small set of labeled training data
- U - unlabeled data

Loop for k iterations:
- Step 1: Train a classifier C_k based on L;
- Step 2: Extract new data D based on C_k;
- Step 3: Add D to L;

At each iteration, the classifier trained on the previous training data (using the features introduced in the previous section) is used to tag the unlabeled data. In addition, for each O token and NE segment, a confidence score is computed using the con-

strained forward-backward algorithm (Culotta and McCallum, 2004), which calculates the L_X^c, the sum of the probabilities of all the paths passing through the constrained segment (constrained to be the assigned labels).

One way to increase the size of the training data is to add all the tokens classified with high confidence to the training set. This scheme is unlikely to improve the accuracy of the classifier at the next iteration because the newly added data is unlikely to include new patterns. Instead, we use the high confidence data to tag other data by exploiting independent features.

- Tagging ORG

  If a sequence of tokens has been classified as "ORG" with high confidence score ($> T$)[1], we force the labels of other occurrences of the same sequence in the same document, to be "ORG" and add all such duplicate sequences classified with low confidence to the training data for the next iteration. In addition if a high confidence segment ends with company suffix, we remove the company suffix and check the multi-mentions of the remaining segment also. In addition to that, we reclassify the sentence after removing the company suffix and check if the labels are still the same with high-confidence. If not, the sequence will be added to the training data. As shown in Example 4, "Safeway shares ticked" is added to training data because "Safeway" has low confidence after removing "Inc.".

  Example 4:
  High-confidence ORG: ***Safeway Inc.*** *shares ticked up ...*
  Low-confidence ORG:
  1) ***Safeway*** *shares ticked up ...*
  2) Wall Street expects ***Safeway*** *to post earnings ...*

- Tagging PER

  If a PER segment has a high confidence score and includes at least two tokens, both this segment and the last token of this segment are used to find their other mentions. Similarly, we force their labels to be PER and add them to the training data if their confidence score is low. However, if these mentions are followed by any company suffix and are not classified as ORG, their labels, as well as the company suffix are forced to be ORG (e.g., Jefferies & Co.). We require the high-confidence PER segment to include at least two tokens because the classifier may confuse single-token ORG with PER due to their common context. For example, "Tesoro proposed 1.63 billion purchase of...", *Tesoro* has high-confidence based on the model, but it represents *Tesoro Corp* in the document and thus is an ORG.

  In addition, the title feature can be used similarly as the company suffix features. If a PER with a title feature has a high confidence score, but has a low score after the title feature is removed, the PER and its neighbors will be put into training data after removing the title-related tokens.

  Example 5:
  High-confidence PER:
  1)Investor AB appoints ***Johan Bygge*** *as CFO...*
  2)He is replacing Chief CEO ***Avallone****...*
  Low-confidence PER:
  1) ***Bygge*** *is employed at...*
  2) He is replacing ***Avallone*** *...*
  (It is obvious for a human-being that Bygge is PER because of the existence of "employed". However, when the training data doesn't include such cases, the classifier just cannot recognize it.)

- Tagging LOC

  The same approach is used for a LOC segment with a high confidence score. We force the labels of its other mentions to be LOC and add them to the training data if their confidence score is low. Again, if any of these mentions follows or is followed by an ORG segment with a high confidence score, we force the labels to be ORG as well. This is because when a LOC is around an ORG, the LOC is usually treated as part of an ORG, e.g., Google China.

[1]Through the rest of the paper, a high confidence score means the score is larger than T. In our experiments, T is set as 0.98. A low confidence score means the score is lower than 0.8.

Example 6:
High-confidence LOC: The former Soviet republic of **Azerbaijan** *is...*
Low-confidence PER:
Azerbaijan *energy reserves better than...*
Change LOC to ORG: shareholders of the ***Chicago Board of Trade****...*

- Tagging O

  Since all the NE segments added to the training data have low confidence scores based on the original model, and especially since many of them were incorrectly classified before correction, these segments form good training data candidates. However, all of them are positive examples. To balance the training data, we need negative examples as well. If a token is classified as "O" with high confidence score and does not have a label feature "O", this token will be used as a negative example to be added to the training data.

Since the features of each token include the features copied from its neighbors, in addition to those extracted from the token itself, its neighbors need to be added to the training set also. If the confidence of the neighbors are low, the neighbors will be removed from the training data after copying their features to the token of interest. If the confidence scores of the neighbors are high, we further extend to the neighbors of the neighbors until low-confidence tokens are reached. We remove low-confidence neighbors in order to reduce the chances of adding training examples with false labels.

Table 3: Step 2 of the semi-supervised algorithm

Step 2: Extract new data D based on C_k
i) Classify kth portion of U and compute confidence scores;
ii) Find high-confidence NE segments and use them to tag other low-confidence tokens
iii) Find qualified O tokens
iv) Extract selected NE and O tokens as well as their neighbors
v) Shuffle part of the NEs in the extracted data
vi) Add extracted data to D

Now we have both negative and positive training examples. However, one problem with the positive data is that the same NE may appear too many times since the multi-mention property is used. For example, the word "Citigroup" may appear hundreds of times in recent financial articles because of the subprime crisis. To account for this bias in the data we randomly replace these NEs. Specifically, we replace a portion of such NEs with NEs randomly chosen from our NE lists. The size of the portion is decided by the ratio of the NEs that are not in our NE list over all the NEs in the gold data.

Table 3 summarizes the key sub-steps in Step 2 of the algorithm. At each step, more non-redundant and high-accuracy data is added into the training set and thus improves the model gradually.

5 Experiments

The data set used in the experiments is explained in Table 4. Although we have 1000 labeled news documents from the Thomson Financial (TF) News source, only 60 documents are used as the initial training data in our algorithm. For the evaluation, the gold data was split into training and test sets as appropriate. The toolbox we used for CRF is Mallet (McCallum, 2002).

Table 4: Data source. Tokens include words, punctuation and sentence breaks.

Gold Data	1000 docs from TF news (around 330 tokens per doc)
Unlabeled Corpus	100,000 docs from TF news

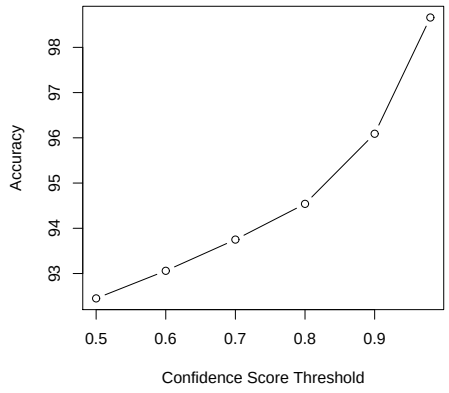

Figure 1: Token accuracy vs confidence score.

We first investigated our assumption that a high confidence score indicates high classification accuracy. Figure 1 illustrates how accuracy varies as CRF confidence score changes when 60 documents

are used as training data and the remaining are used as testing data. When the threshold is 0.98, the token accuracy is close to 99%. We believe this accuracy is sufficiently high to justify using the high confidence score to extract tokens with correct labels.

Table 5: Precision and recall of the automatically extracted training data

NE	Precision%	Recall%	F-score%
LOC	94.5	96.8	95.6
ORG	96.6	93.4	94.9
PER	95.0	89.6	92.2

We wished to study the accuracy of our training data generation strategy from how well it does on the gold data. We treat the remaining gold data (except the data trained for the initial model) as if they were unlabeled, and then applied our data extraction strategy on them. Table 5 illustrates the precision and recall for the three types of NEs of the extracted data, which only accounts for a small part of the gold data. The average F-score is close to 95%. Although the precision and recall are not perfect, we believe they are good enough for the training purpose, considering that human tagged data is seldom more accurate.

We compared the semi-supervised algorithm with a supervised algorithm using the same features. The semi-supervised algorithm starts with 60 labeled documents (around 20,000 tokens) and ends with around 1.5 million tokens. We trained the supervised algorithm with two data sets: using only 60 documents (around 20,000 tokens) and using 700 documents (around 220,000 tokens) respectively. The reason for the choice of the training set size is the fact that 20,000 tokens are a reasonably small amount of data for human to tag, and 220,000 tokens are the amount usually used for supervised algorithms (CoNLL 2003 English NER (Sang and Meulder, 2003) training set has around 220,000 tokens).

Table 6 illustrates the results when 300 documents are used for testing. As shown in Table 6, starting with only 6% of the gold data, the semi-supervised algorithm achieves much better results than the semi-supervised algorithm when the same amount of gold data is used. For LOC, ORG, and PER, the recall increases 5.5, 16.8, and 8.2 respectively, and the precision increases 2.4, 1.5, and 6.8 respectively. Even compared with the model trained with 220,000 tokens, the semi-supervised learning algorithm is better. Especially, for PER, the precision and recall increase 2.8 and 4.6 respectively. Figure 2 illustrates how the classifier is improved at each iteration in the semi-supervised learning algorithm.

Table 6: Classification results. P/R represents Precision/Recall. The numbers inside the parentheses are the result differences when the model trained from 60 docs is used as baseline.

Training Data	P/R(LOC)	P/R(ORG)	P/R(PER)
60 docs	88.1/85.6	86.0/64.2	74.5/81.2
700 docs	91.2/88.2 (3.1/3.6)	90.5/76.6 (4.5/12.4)	78.3/84.8 (3.8/3.6)
semi-supervised (60 docs)	90.5/91.1 **(2.4/5.5)**	87.5/81.0 **(1.5/16.8)**	81.1/89.4 **(6.6/8.2)**

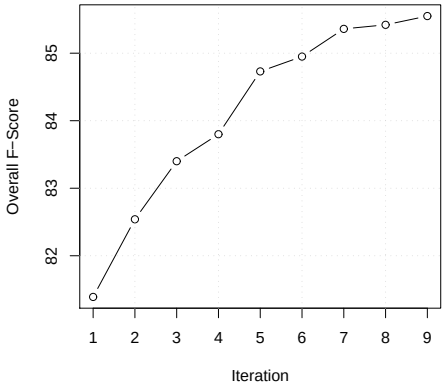

Figure 2: Overall F-score vs iteration numbers

Table 7 compares the results when the multi-mention property is also used in testing as a high-precision rule. Comparing Table 7 to Table 6, we can see that with the same training data, using multi-mention property helps improve classification results. However, this improvement is less than that obtained by using this property to extract training data thus improve the model itself. (For a fair comparison, the model used in the semi-supervised algorithm in Table 6 only uses multi-mention property to extract data.)

Our last experiment is to test how this method can be used when the initial gold data and the testing data are from different domains. We use the CoNLL 2003 English NER (Sang and Meulder, 2003) training set as the initial training data, and automatically extract training data from the TF financial news corpus. The CoNLL data is a collection of news wire

documents from the Reuters Corpus, while TF data includes financial-related news only. Table 8 illustrates the results. As shown in the table, with only CoNLL data, although it contains around 220,000 tokens, the results are not better than the results when only 60 TF docs (Table 6) are used for training. This indicates that data from different domains can adversely affect NER accuracy for supervised learning. However, the semi-supervised algorithm achieves reasonably high accuracy. For LOC, ORG, and PER, the recall increases 16, 20.3, and 4.7 respectively, and the precision increases 4.5, 5.5, and 4.7 respectively. Therefore our semi-supervised approach is effective for situation where the test and training data are from different sources.

Table 7: Classification results when multi-mention property (M) is used in testing

Trainig Data	P/R(LOC)	P/R(ORG)	P/R(PER)
60 docs +M	89.9/87.6	82.4/71.4	78.2/87.3
700 docs+M	91.2/89.1 (1.3/1.5)	90.2/78.3 (7.8/6.9)	79.4/91.1 (1.2/3.8)
semi-supervised +M (60 docs)	90.0/91.0 (1.1/3.4)	86.6/82.4 (4.2/11.0)	81.3/90.6 (3.1/3.3)

Table 8: Classification results trained on CoNLL data and test on TF data. Training data for the semi-supervised algorithm are automatically extracted using both multi-mention and high-precision context from TF corpus.

Training Data	P/R(LOC)	P/R(ORG)	P/R(PER)
CoNLL	85.6/74.7	75.2/65.9	72.4/85.2
Semi-supervised (CoNLL)	90.1/90.7 **(4.5/16)**	81.7/86.2 **(5.5/20.3)**	77.1/90.5 **(4.7/4.7)**

6 Conclusion

We presented a simple semi-supervised learning algorithm for NER using conditional random fields (CRFs). In addition we proposed using high precision label features to improve classification accuracy as well as to reduce training and test time.

Compared to other semi-supervised learning algorithm, our proposed algorithm has several advantages. It is domain and data independent. Although it requires a small amount of labeled training data, the data is not required to be from the same domain as the one in which are interested to tag NEs. It can be applied to different types of NEs as long as independent evidence exists, which is usually available. It is simple and, we believe not limited by the choice of the classifier. Although we used CRFs in our framework, other models can be easily incorporated to our framework as long as they provide accurate confidence scores. With only a small amount of training data, our algorithm can achieve a better NE tagging accuracy than a supervised algorithm with a large amount of training data.

References

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. *Proceedings of the Workshop on Computational Learning Theory*, pages 92–100.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

A. Culotta and A. McCallum. 2004. Confidence estimation for information extraction. *HLT-NAACL*.

G. D. Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Feng Jiao, Shaojun Wang, Chi H. Lee, Russell Greiner, and Dale Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics*, pages 209–216, July.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. *CoNLL*.

A.K. McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Andrew McCallum. 2003. Efficiently inducing features of conditional random fields.

David Pierce and Claire Cardie. 2001. Limitations of co-training for natural language learning from large datasets. In *EMNLP*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *CoNLL*, pages 142–147.

Yingchuan Wong and Hwee Tou Ng. 2007. One class per named entity: Exploiting unlabeled text for named entity recognition. *IJCAI*, pages 1763–1768.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Meeting of the Association for Computational Linguistics*, pages 189–196.